(12)

R and D CENTER LABORATORY

TECHNICAL REPORT

NO. 12907

IMAGE ACQUISITION AND MANIPULATION SYSTEM

JANUARY 1984

by FRANCIS B. HOOGTERP
STEVEN A. CAITO
US Army Tank-Automotive Command
ATTN: DRSTA-ZSC
Warren, Michigan 48090

U.S. ARMY TANK-AUTOMOTIVE COMMAND
RESEARCH AND DEVELOPMENT CENTER
Warren, Michigan 48090

NOTICES

This report is not to be construed as an official Department of the Army position.

Mention of any trade names or manufacturers in this report shall not be construed as an official indorsement or approval of such products or companies by the US Government.

Destroy this report when no longer needed. Do not return it to the originator.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 12907	2. GOVT ACCESSION NO. AD-A138927	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) IMAGE ACQUISITION AND MANIPULATION SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Francis B. Hoogterp Steven A. Caito		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS US Army Tank-Automotive Command R&D Center (DRSTA-ZSC) Warren, Michigan 48090		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE January 1984 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image Processing
Digitization
Image Restoration
Filtering
Noise

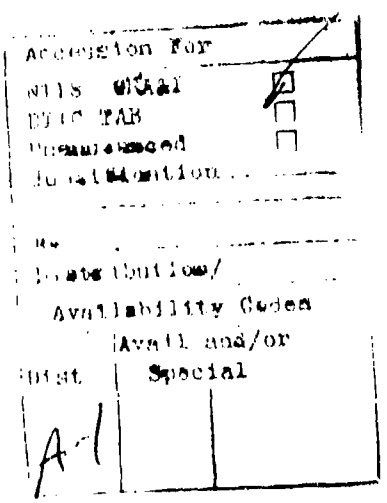
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A complete image processing system has been developed on a Hewlett Packard (HP) 1000 Series F Computer. This system is interfaced with a digital graphics CBX800 image frame grabber and display system. This report briefly describes the hardware interface required between the HP and the CBX. A more complete description is provided of the Image Processing Library. The Image Processing Library (developed explicitly for the system) consists of two levels. Interactive programs are included that provide a user with immediate access to

standard image processing capabilities. Also, a FORTRAN callable library of image processing subroutines is described to speed the development of exploratory image processing algorithms.

TABLE OF CONTENTS

Section		Page
1.0.	INTRODUCTION	5
2.0.	OBJECTIVE	5
3.0.	CONCLUSIONS	7
4.0.	RECOMMENDATIONS	7
5.0.	DISCUSSION	7
5.1.	Useage as Stand-Alone Image Processor	7
5.2.	Image Processing Subroutine Library: General Usage	15
5.3.	Image Processing Library: Subroutine Descriptions	20
APPENDIX A.	HP-CBX Interface and Driver Description	A-1



LIST OF ILLUSTRATIONS

Figure	Title	Page
1-1.	System Configuration	6
5-1.	Image Menu	11
5-2.	Parameters	13

1.0. INTRODUCTION

IMAG is an image processing package designed to be run on an HP 1000 Series F digital computer interfaced with a CBX800 Imaging System. The software package has the ability to digitize an image, as well as store the image on disk or tape. A block diagram of the system configuration can be seen in Figure 1-1. The HP computer is the focus of the system. An image, as seen by a standard closed-circuit television camera, is digitized by the CBX800 and held in its memory. With the proper commands from the HP computer, the image is passed from the CBX800 to the HP via a 8 bit wide parallel interface, and then stored on disk or, if desired, stored on tape. Conversely, if you wish to display a stored image, you must first read the image from disk or tape. The image existing in the HP computer is then sent to the CBX800 where it is displayed on the color monitor. A number of operations may be performed on an image in the HP computer. After a particular image has been filtered, convoluted, histogram equalized, etc., it must be sent back to the CBX800 if the modified image is to be seen. A number of images may be displayed simultaneously, depending on the size of each image.

This report is a brief summary on the use of the aforementioned system, along with the necessary information to enable the user to fabricate his/her own programs.

2.0. OBJECTIVE

This report is intended to assist the user in the operation of the Hewlett-Packard (HP) CBX800 computer system. It is not intended to be a comprehensive manual on the HP operating system. Additional questions pertaining to this subject should be directed to the system manager. A complete set of manuals are supplied by Hewlett-Packard for your particular system.

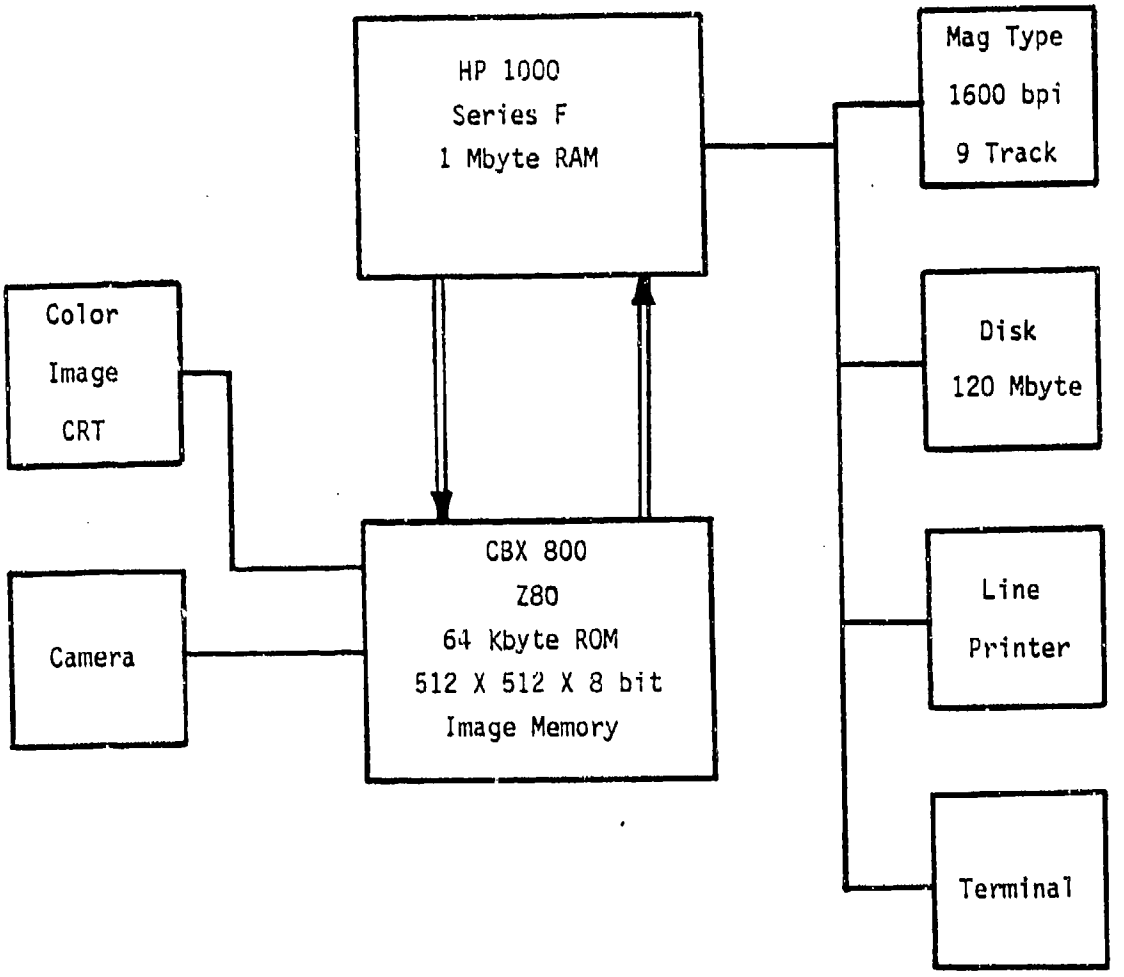


Figure 1-1. System Configuration

3.0. CONCLUSIONS

This report has no conclusions. It is intended to assist the user in the operation of the Hewlett-Packard CBX800 computer system.

4.0. RECOMMENDATIONS

This report makes no recommendations, for the reason given above.

5.0. DISCUSSION

5.1. USAGE AS STAND-ALONE IMAGE PROCESSOR

Two separate, interactive programs have been developed on the HP to provide many of the capabilities of a stand-alone image processing system. This section includes a discussion of how to log on to the HP and subsequently, on the use of the two programs, DIGIT and IMAG, that comprise the image processor capability.

When you sit down at the terminal, the computer requires you to identify yourself. This is accomplished by the log-in procedure. If the person before you has logged off, when a key is pressed (any key), the HP computer will respond with

PLEASE LOG-ON:

Press any key. When you obtain the response above, type in your identification code. (This is ususally your name, followed by a period, followed by your group code.)

Example:

PLEASE LOG-ON:JANE.ENG

If you don't have an identification code, see the system manager to obtain one. If the above is executed properly, the system will respond with

PLEASE LOG-ON:JANE.ENG
PASSWORD?

The computer is now looking for a sequence of characters (numbers or letters) which are known, or should be known, only by you. The password, in effect, is your "key" to your files. This is your only protection against unfriendly forces who wish to acquire your programs and/or files. This password may be changed at any time by use of the system commands.

Type in your password

PLEASE LOG-ON:JANE.ENG [Press Return]
PASSWORD? [Press Return]

When typing in your password, you will not see it appear on the screen. This is an added protection against wandering eyes. If the above has been done correctly, the system will identify itself to you, and display its messages. You are now logged on. If done incorrectly, the system will respond with an error message.

Example:

SESSION 1 LGON 05 ILLEGAL ACCESS
UNABLE TO COMPLETE LOG-ON

After displaying the system messages, the computer will prompt you with a colon (:). You are now ready to use the system as an image processing station. Before the computer can process an image, the image must be digitized and stored in the HP's memory. The program DIGIT was designed to expedite the digitizing and saving of desired images.

Setting up the system to digitize an image requires a closed-circuit TV camera. The output of the camera (video output) must be connected through a B-C connector to the CAM input on the CBX800. The connectors are located on the back of the device. The system is now ready to run the digitize program.

To run a program, you must use the system command RU. This command is followed by a comma and the name of the program. In this case, we want to run the digitizer; therefore, we would type in the following:

RU,DIGIT [Press Return]

The system will now execute the program. The program will immediately describe itself by stating:

> THIS PROGRAM ALLOWS THE DIGITIZATION OF AN IMAGE FROM A CAMERA INPUT. THE IMAGE IS READ BY THE HP, CONVERTED TO PIXEL FORM, AND STORED ON THE DISK. THE IMAGE MAY ALSO BE READ FROM THE DISK AND DISPLAYED.

It will then inquire whether you wish to initialize the CBX800. If the CBX800 has not been used prior to this program call, it must be initialized.

INITIALIZE CBX (YES=Y)?

Type YES [Press Return]

If CBX initialization is attempted, the computer may respond with

PUSH CBX RESET BUTTON & RETURN (TYPE -1 TO ABORT)

In this case, simply push the RESET button on the front panel of the CBX and press the return key on the terminal. Initialization will black out the color monitor. Regardless of whether or not the CBX is initiated, the program will then provide a list of options as follows:

OPTIONS

QUIT	= QI
ERASE	= ER
DIGITIZE & SAVE	= DS
READ DISK IMAGE	= RD
READ CBX IMAGE	= RC
DISPLAY IMAGE	= DI
SAVE IMAGE	= SI

QI - Terminate program execution
ER - Erase color monitor (clear CBX memory)
DS - Digitize image on color monitor, store image in CBX memory, and save image on disk under file name specified.
RD - Read image from disk file (image exists in HP)
RC - Read image from CBX memory, place contents in HP memory
DI - Display image in HP memory on the color monitor
SI - Save image in HP memory on disk

When the option menu first appears, the color monitor will have a green hue. Whatever you point the camera at will appear on the monitor. The monitor is said to be "live", or in direct video mode. If you choose to digitize and save an image (i.e. select the "DS" option), the program will respond with

IMAGE SIZE & LOCATION LOCATION (NX,NY,IXLOC,IYLOC)

The CBX has the capacity to accept an image which is 512 x 512, that is 512 pixels across and 512 pixels high. Each pixel is represented by 8 bits that must be stored in the CBX memory and ultimately in the HP memory with a final destination of the hard disk. The program presently is designed to handle an image of 256 x 256. Note: NY & NY may take on values in specified range -- NX = 1-256, NY = 1-256.

IXLOC and IYLOC are the coordinates of the lower left-hand corner of the image you want to digitize and store. The screen is laid out with x=0, y=0 being the lower left-hand corner of the screen with x increasing as you move to the right, and y increasing as you move upward. When you have selected these parameters, white boundaries will appear on the screen and again a list of options will appear.

ABORT = AB
DIGITIZED VIEW = DV
LIVE SCREEN = LS
SAVE IMAGE = SI

Before you save your image, make last minute adjustments on position, focus, and aperture settings. When the digitized view is selected the white boundary lines will disappear and the green screen will become black and white, shades of gray. The program will again return with a list of options as before. This permits switching between the direct video image and the digitized form to further aid in making adjustments. It should be noted that selecting the digitized view option "DV" only affects the image in the CBX memory and not the image in the HP memory. The digitized image is placed in the HP memory (and on the disk) when the save image option is used. If you choose to save the image, the program will prompt you for a file name:

ENTER IMAGE FILE NAME

When the file name has been entered, the program will return with

ENTER IMAGE INDENTIFICATION

You may now enter a short sentence describing your image. Don't become impatient. The length of time between commands depends on the size of the image.

A stand-alone image processor is also expected to offer a variety of image manipulation, image restoration, and image enhancement capabilities. These fuctions may be invoked for the interactive environment, provided by the program IMAG.

IMAG is the second program developed as part of the stand-alone imaging system and serves as the controller to all the subroutines which perform the actual number crunching on the data (image). This program can be executed with the RU, command by typing

RU,IMAG [Press Return]

The program will respond by asking if you wish to initialize CBX just as the DIGIT program does. The response to this query is handled exactly as it is for DIGIT.

Once the initialization is completed or bypassed, IMAG should display a menu of all possible options available. An example can be seen in Figure 5-1.

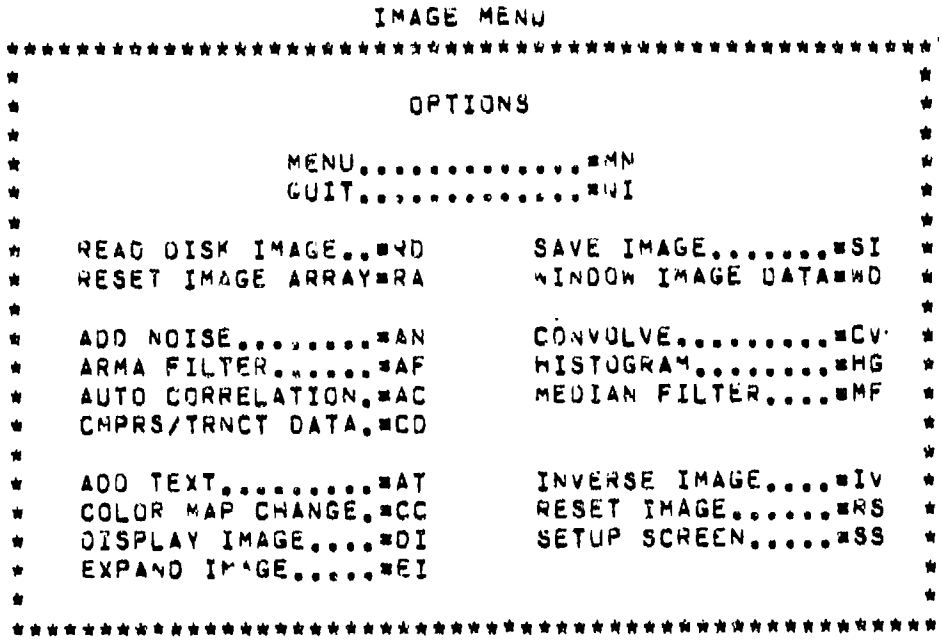


Figure 5-1. Image Menu

The option list is segmented into four basic sections. The first section contains the most essential commands to the operation of the program. The second set pertains to accessing of image data, setting bounds on the data, and the storing of altered data. The commands for image processing are located in the third set. This is where noise can be added and various types of filtering algorithms can be selected. The last set of commands are those which alter the color monitor in some way. Text may be added, as well as various types of color maps. After the menu has appeared, the program will prompt you with the statement:

PLEASE INPUT AN OPTION OR TYPE -MN- FOR MENU

This prompt will appear after the completion of every command in the menu except the QUIT command. If the above prompt does not appear, you are still in a sub-level of that particular command. If the command QUIT is chosen, the HP will respond with:

IMA01 STOP 0000

The HP system prompt will again appear (:), signaling to the user that the operation of IMAG has been terminated.

As a sample of the program's operation, let us run through a typical sequence of commands. After the program prompt has appeared:

PLEASE INPUT AN OPTION OR TYPE -MN- FOR MENU

Type SS [Press Return]

IMAG will respond with:

ENTER BACKGROUND COLOR (R,G,B)

IMAG is now looking for three numbers, separated by commas, between 0 and 252 inclusive. Each number represents the intensity of the red, green, and blue gun of the color monitor, 0 being lowest intensity and 252 being the brightest. When the background color has been entered, IMAG will ask for window color, and lastly, for text color. Default values for background and window color are (0,0,0), black. The default values for the text color are (252,252,252), white. IMAG will then prompt the user for image window size and position. If no values are entered, they will take on the values of the image size and position. Default values can be obtained by pressing the return key after each question. The screen setup routine will then ask if

you wish to change it. If you press return, it will remain as it was. After the final input has been entered, IMAG will set up the screen in the colors/ options you desire and return to the program prompt.

Before one can do image processing, one must have an image. Therefore, type:

RD [Press Return]

IMAG will respond with:

ENTER IMAGE FILE NAME

Here the program is looking for a particular file name. If IMAG cannot find your image, it will give you an error message and inquire again for a file name.

Type /TANK [Press Return]

IMAG will return with

```
IMAGE ID...TANK FIRING
IMAGE SIZE (256X256)
CHANGE ISTRT,JSTRT & NX,NY OR RETURN
```

In these statements, IMAG is telling you what file was read, the size of the file, in pixels, and allowing you to select only a part of the image, if that is desired. If you press return, nothing will be altered. In Figure 5-2 below are the parameters which designate how much of the image you wish to read in.

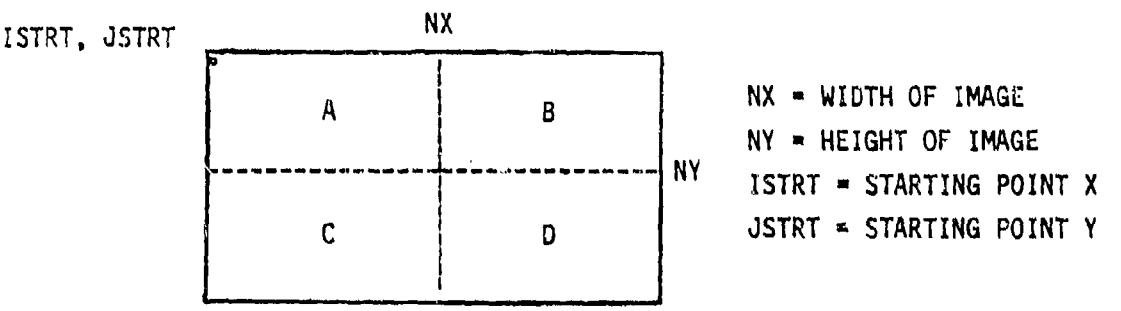


Figure 5-2. Parameters

In this way, you may read in any part of the image. For example, if we wish to read in the whole image (default) ISTRT=1, JSTRT=1 and NX=256, NY=256. If we wish to read in section B only ISTRT=128, JSTRT=1 and NX=128, NY=128 and so on. ISTRT and JSTRT must be positive and the following

inequalities must be adhered to when selecting the parameter values:

ISTRT + NX image file NX
JSTRT + NY image file NY

To display the image,

Type D1 [Press Return]

The program will return with the statement:

DESTINATION IMAGE LOCATION (XPOS,YPOS)

Type 0,225 [Press Return]

Once the image is read into the HP memory, it is ready to be processed. To discover the effects of filtering, it may be desirable to first create an image which is something less than perfect.

PLEASE INPUT AN OPTION OR TYPE -MN- FOR MENU.

Type AN [Press Return]

This routine will add noise to any image contained in the HP memory. It will respond with:

MEAN & STANDARD DEVIATION

IMAG is now looking for two numbers which describe the statistics of the Gaussian white noise added to the image in the HP memory.

Type 0,4.4 [Press Return]

When IMAG returns, we want to display the noisy image.

PLEASE INPUT AN OPTION OR TYPE -MN- FOR MENU.

Type DI [Press Return]

The program will respond with:

DESTINATION IMAGE LOCATION (XPOS,YPOS)

Type 256,225 [Press Return]

By adding noise to your image you may have set your individual pixel intensities out of the valid range of 0 thru 252; therefore, it may be necessary to truncate or linearly compress your image. If so, IMAG will detect this and ask to remedy the situation. Type in your choice and press return. You are now ready to filter your image using any desired combination of options. For example, a 3 x 3 convolution or masking filter can be employed by typing:

CV [Press Return]

The convolve routine will prompt you to enter the elements of your matrix. The maximum dimensions of the matrix are 3 x 3. Smaller matrices may be realized by setting rows or columns equal to zero. The elements must be entered by rows. In this case, type in a matrix of all ones.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

It must always be remembered that the image enhancement and restoration options affect only the image that is in the HP memory and not the image on the monitor (and in the CBX memory). The modified image must be specifically displayed if it is to be seen. When completed, the prompt will again appear and you may add text, and change color maps without effecting the structure of the image. If a mistake is made and you wish to start again, the screen may be reset with the RS command.

Additional details regarding the various menu options are contained in the individual subroutine descriptions of the following section.

5.2. IMAGE PROCESSING SUBROUTINE LIBRARY: GENERAL USAGE

If the user has the ability to program in FORTRAN, he/she is not limited to IMAG. All the subroutines listed in the next section are FORTRAN callable. If the user has a special application, he may tailor a program for his particular needs.

To amalgamate your custom programs with those of the system library's, you must use the command file ".LD". This file is required when loading a program after it has been compiled. It contains all the necessary instructions to attach the subroutines in the library to your custom program. To load your program after it has been compiled, use the instruction:

```
LOADR,.LD,%NAME
        (your program name)
```

If no undefined externals result, you are ready to run your program. If undefined externals do exist, you need to include your subroutine file names in with the others in the .LD file. Use the SEARCH command.

Example:

```
SE%SUBNM
      (your subroutine name)
```

Insert this command at the top of your own version of the .LD command file.

The system library consists of a number of FORTRAN callable subroutines. Because of the large amounts of data that must be passed from subroutine to subroutine, common blocks must be utilized in accomplishing the task. The first common block is CXPIX which contains 65,536 bytes of information used to store pixel intensities. If a subroutine is developed which uses this data, it must be declared common at the beginning of your routine. The syntax of the statement is as follows:

```
COMMON/CXPIX/IPIX(256,256)
```

The second common block is CXBUF. The use of this common block is dependent upon the subroutines or groups of subroutines you are working with.

If the user wishes to create his own common blocks, he must do so by use of the following statements:

```
BLOCK DATA SAMPL
COMMON/NAME/PAR1,PAR2,PAR3(50)
COMMON/NAME2/PAR4,PAR5(50),...
.
.
.
END
```

These statements comprise a separate program segment and should appear following the END statement of any one of the newly created subroutines.

IMAGE MANIPULATION ROUTINES

These routines operate on images in pixel format.

SETSC	-	permits standard screen layout to be set up
IMGNS	-	adds Gaussian Noise to an image
CONME	-	performs an arbitrary 3 x 3 convolution on the image or median filters the image
HIST	-	performs histogram stretching or thresholding
OBSER	-	performs a 2-D ARMA filtering of an image
BLANK	-	sets image array to a specified intensity
CBAR	-	draws a color bar on the screen
INVRT	-	replaces the image array with its inverse video
WIND	-	permits definition of image array window
STRIP	-	performs general strip filtering of image array
MATUP	-	performs updates on matrix rows and columns
CXMAP	-	redefines color maps and defines submaps

IMAGE TRANSFER ROUTINES USING THE CBX WINDOW FORMAT

CBXDP - reads an image from the disk and writes the image to the CBX
CBXDG - digitizes the full camera image into the CBX memory, reads and stores the image on the disk

IMAGE TRANSFER ROUTINES FOR IMAGES IN PIXEL FORMAT

IMAGE - outputs image in HP meory to CBX memory (to CRT)
CXIMG - reads image from CBX memory and places in HP memory
CBXGI - reads image from disk into HP memory
CXSAV - writes image currently in HP memory, onto the disk
CBXDZ - digitizes an image from a camera and places image in CBX memory, HP memory, and optionally on the disk
LIMAG - uses linear interpolation to display double image
DIMAG - uses pixel replication to display double image

LOW LEVEL CBX MANIPULATION ROUTINES

CBCAM - turns camera on or off
BOX - draws and fills a rectangular box with the specified color
IPLOT - draws a visible or invisible vector
TEXT - prints text on a screen
COLOR - selects a color for alpha or vector output
ERASE - fills the screen with a given color
CXINT - initializes the CBX
STATS - prints CBX status
CBXAD - converts screen coordinates to CBX format
CBXCM - outputs one command (with parameters) to CBX
WINDO - reads/writes one 8K image window from/to CBX
CBXSM - select or load a color map
WAIT - causes HP to wait a specified time

IMAG

IMAG is the main line program which draws on all the following subroutines. It has the ability to process any one image in a number of different ways and then display the image on a CRT. The maximum image size which it can handle is 256 X 256.

The image processing techniques include adding noise, convoluting the image with any 3 X 3 matrix, equalization, thresholding, performing autocorrelations, expanding, ARMA filtering, median filtering, and displaying the image in one of four color maps or in one specified by the user. Text can also be written on the screen to better explain an image.

SUBROUTINE CALLS: AUTOC, BLANK, CBXGI, CBSXM, CONME, CXMAP, CXSAV, HIST, IMAGE, IMGCK, IMGNS, LIMAG, SET, STRIP, TEXT, WIND

SOURCE CODE: &IMAG

5.3 IMAGE PROCESSING LIBRARY: SUBROUTINE DESCRIPTIONS

The following pages contain a list of subroutines arranged alphabetically.

SUBROUTINE AUTOC

This subroutine will calculate the autocorrelation coefficients of a specified image stored in the IPIX array. Horizontal and vertical lags are entered interactively during the program.

CALLING SEQUENCE: CALL AUTOC (ISTRT, JSTRT, N, M)

WHERE:

ISTRT: The first point in the X direction for which the autocorrelation calculations will begin for a specific image.

JSTRT: The first point in the Y direction for which the autocorrelation calculations will begin for a specific image.

N: The horizontal length of the window in which autocorrelation calculations will take place.

M: The vertical length of the window in which autocorrelation calculations will take place.

SUBROUTINE CALLS: None

SOURCE CODE: &AUTOC

SUBROUTINE BLANK

This subroutine will set an entire image stored in the IPIX array to a blank image with all pixels equal to the intensity level specified.

CALLING SEQUENCE: CALL BLANK (NX, NY, ICOL)

WHERE:

NX: Number of points in x direction

NY: Number of points in y direction.

ICOL: The desired intensity level.

SUBROUTINE CALLS: None

SOURCE CODE: &BLANK

SUBROUTINE BOX

This routine draws and fills a rectangular box of specified color. The current cursor position is used as the upper left corner of the rectangle and the desired location of the lower right corner is specified. Upon completion, the cursor is positioned at the lower right corner of the rectangle.

CALLING SEQUENCE: CALL BOX (IX, IY, ICOL)

WHERE:

IX, IY: Desired coordinates for lower right corner of the rectangle.

ICOL: Desired intensity of rectangle.

SUBROUTINE CALLS: COLOR, CBXAD, CBXCM

SOURCE CODE: &SBR

SUBROUTINE CBAR

This routine draws a color bar (or grayscale bar) on the screen at the specified position. A total of 17 levels are shown from 0 thru 252.

CALLING SEQUENCE: CALL CBAR (IX, IY)

WHERE:

IX, IY - Lower left corner of color bar, note the color bar is 340 by 20 pixels.

SUBROUTINE CALLS: IPLOT, BOX

SOURCE CODE: &SET

NOTE: The color bar consists of 17 squares placed in a horizontal row. each 20 X 20 pixels of a particular intensity. The first square has intensity 0 and the intensity for each subsequent square is incremented by 16 (except the last which is limited to 252). When the negative grayscale map (color may 1) is used, the color bar is inverted, with alternate color map schemes the color bar is altered accordingly.

SUBROUTINE CBCAM

This routine will turn the direct video capability of the CBX on or off as specified. When the direct video is enabled the CRT will display the output of the TV camera with no intervention from the CBX. In this case the sync signal is also set to external and is supplied by the camera. When the direct video is disabled (turned off) the CRT displays the image held in the CBX memory. In this case the sync signal is also supplied by the CBX and the camera has no influence on the CRT or the CBX.

CALLING SEQUENCE: CALL CBCAM (ION)

WHERE:

ION = 0 to disable camera

= 1 to enable direct video output from camera to CRT

SUBROUTINE CALLS: CMXCM

SOURCE CODE: &SBR

SUBROUTINE CBXAD

This routine is called by various imaging routines and converts a set of screen coordinates into the CBX required format. The screen coordinates vary from 0 thru 511 in the x direction and from 0 thru 479 in the y direction. Since the CBX data interface is only 8 bits wide each coordinate is separated into a low order byte and a high order byte. The coordinates are then placed in the CBX command transfer buffer (common block /PARMS/) in the following order; low order x, low order y, and high order y.

CALLING SEQUENCE: CALL CBXAD (IX, IY)

WHERE:

IX - desired x coordinate

IY - desired y coordinate

SUBROUTINE CALLS: None

SOURCE CODE: &SBR

NOTE: This routine is generally not called directly by a user program.

SUBROUTINE CBXCM

This routine is called by various imaging routines to send commands to the CBX. Upon entry to this routine the common block /PARMS/ contains the desired CBX command and parameters. The HP privileged status mode is turned on by the CPX driver when it is called to transmit the command and any required parameters. The privileged status is then turned off.

CALLING SEQUENCE: COMMON/PARMS/NCMD, NPAR, IPAR (21)

WHERE:

NCMD - CBX command number

NPAR - number of parameters (0-21)

IPAR - array containing parameters

SUBROUTINE CALLS: EXEC, PVON, COMND, PVOFF

SOURCE CODE: &SBR

NOTE: This routine is usually not called directly from user program.

SUBROUTINE CBXDG

This routine causes the CBX to digitize a full 512 x 512 pixel by 8 bit image from a camera input. The HP reads full image in CBX format and stores the image in the specified disk file. The routine interactively requests the file name and image identification label from the user. The camera is connected live to the CRT to allow camera adjustments. When the user is ready to digitize, the "RETURN" key is depressed.

CALLING SEQUENCE: CALL CBXDG

SUBROUTINE CALLS: OPEN, WRITE, CBCAM, CBSCM, WINDO, CLOSE

SOURCE CODE: &CXSBI

NOTE: The data format generated by this routine is not compatible with pixel manipulation routines. This routine does provide very fast image retrieval and display, however, taking only about 15 seconds for a full 512 x 512 x 8 bit image.

SUBROUTINE CBXDP

This routine reads a disk file created by the digitizing routine "CBXDG", and displays the image data on the CRT. The disk file name is requested interactively and the image identification label from that file is printed on the user's terminal. The complete 512 x 512 pixel x 8 bit image is then output to the CBX. The reading of the disk file and displaying of the image on the CRT take about 15 seconds.

CALLING SEQUENCE: CALL CBXDP

SUBROUTINE CALLS: OPEN, READF, WINDO, CLOSE

SOURCE CODE: &CXSBI

SUBROUTINE CBXDZ

This routine causes the CBX to digitize an image from a camera input. The HP then reads the image from the CBX, converts the image to pixel form and if requested, saves the image on the disk. The user is interactively asked the size of the requested image in pixels and its location on the CRT. A rectangle is then drawn on the screen around the desired image area. The user can then adjust the camera while viewing the live camera output or a digitized output. When the user requests that this image be saved, the camera signal is digitized by the CBX, read into the HP, converted into pixel format and placed in IPIX array. If the user has also requested that the image be stored on the disk, "CXSAV" is called to perform this task. The routine is called with the following statements:

CALLING SEQUENCE: COMMON /CXPIX/IPIX (256, 256)

.
.
.

CALL CBXDZ (NX, NY, IDIG)

WHERE:

IPIX - array of image pixel intensities

NX - number of pixels per line in digitized image

NY - number of lines of pixels in digitized image

IDIG - if IDIG = 1 the digitized image will also be saved on the disk

SUBROUTINE CALLS: CBXCM, CBCAM, COLOR, ERASE, IPLOT, CXIMG, CXSAV

SOURCE CODE: &CXSBI

SUBROUTINE CBXGI

This routine reads a pixel oriented image from the specified disk file into the IPIX array of the HP memory. The disk file must be compatible with the format produced by "CXSAV". The file name is requested interactively and the image identification label is printed on the user's terminal. The user can also interactively request that only part of the image be read in.

CALLING SEQUENCE:

```
COMMON/CXPIX/IPIX (256, 256)
      .
      .
      .
CALL  CBXGI (NX, NY)
```

WHERE:

IPIX - the image array
NX - number of pixels per line
NY - number of lines of pixels

SUBROUTINE CALLS: OPEN, READF, CLOSE

SOURCE CODE: &CXSBI

SUBROUTINE CBXSM

This routine allows the user to modify a color map or, alternatively, to select which color map is to be used with the currently displayed image.

The CBX system has four separate color maps (numbered 0-3) of which any single one can be made active at any time (map no. 0 is the default active map). A color map is comprised of three separate 256 element arrays. These three arrays represent the amount of red, the amount of green, and the amount of blue respectively for each of the 256 possible intensity values. The CBX uses the intensity of each pixel as an index into each of the three color arrays of the currently active color map to find the appropriate intensity for each color at that pixel. For example, if a certain pixel's intensity is 105, then the 105th element of the first array in the active color map will specify the amount of red at that pixel. The 105th element of the second array will be the intensity of blue for that pixel.

CALLING SEQUENCE: COMMON/CXBUF/ICOL (3256), IBUF (3328)

.
.
.

CALL CBXSM (MAP, ISET, NLOC)

WHERE:

MAP	- the desired map number (0-3)
ISET	- specifies desired function =0 make the specified map active >0 load the first ISET colors into the specified map from the array ICOL
NLOC	- number of locations one wishes to change
ICOL (I,J)	- this array contains the desired color map

SUBROUTINE CALLS: CBXCM

SOURCE CODE: &SBR

SUBROUTINE CONME

This routine will perform a general 3 x 3 convolution on the image array in the HP memory. The convolution weighting matrix is passed by row in one-dimentional floating point array elements. The original image, contained in the IPIX array is overwritten by the convolved image. Required pixels not included in the original image are assumed to have the same intensity as the closest pixel within the image. This routine will also compute the median of a 3 x 3 matrix of data. That is, it finds the middle value, i.e., the value which has as many values (intensities) greater than itself as values lesser than itself and loads this intensity in the IPIX array, therefore, the new array will contain the middle value of its eight immediate neighbors.

CALLING SEQUENCE: COMMON / CXPIX/IPIS (256, 256)

.
.
.

CALL COME (ISTRT, JSTRT, N, M, XMAT, IOPU)

WHERE:

(ISTRT, JSTRT)	- starting location in image array, normally (1,1)
(N,M)	- number of image pixels in horizontal and vertical directions respectively
XMAT	- matrix of convolution coefficients
IOPT1	- desired function convolution = 0 median = 6

NOTE: This routine does not alter the image on the screen.

The XMAT array contains 9 elements to be multiplied by the 8 nearest pixels to each pixel. The 9 products are summed and divided by the sum of the XMAT array (or by 1 if XMAT sum is 0). The resulting value replaces the original pixel value.

SUBROUTINE CALLS: None

SOURCE CODE: &CONME

SUBROUTINE CXIMG

This routine reads the image data from the CBX and extracts the specified portion of the image. The extracted data is converted to pixel form and placed in the IPIX image array.

CALLING SEQUENCE: CALL CXIMG (NX, NY, IX, IY)

WHERE:

NX - desired number of pixels per image row

NY - desired number of rows of pixels

IX - x coordinate of lower left corner of desired image

IY - y coordinate of lower left corner of desired image

SUBROUTINE CALLS: WINDO

SOURCE CODE: &CXSBI

SUBROUTINE CXINT

This routine initializes the CBX routine called after powering up the CBX and CRT. The CBX is initialized for a 512 x 480 image with 8 bits per pixel. The color maps are initialized to their default values and the screen is blackened (set to all zeros). The color is then set to white (255) in order that subsequent output of vectors or text can be seen.

CALLING SEQUENCE: CALL CXINT (MODE)

WHERE:

MODE - currently has no effect

SUBROUTINE CALLS: PVON, CXSTS, PVOFF, CBXCM, ERASE, COLOR

SOURCE CODE: &SBR

NOTE: Whenever this routine is called, whatever was in the CBX memory (and on the CRT) is lost. It is not required to call CXINT each time a program is run. It is only required that his routine be called once after powering up the CBX and CRT.

SUBROUTINE CXMAP

This routine provides the capability to interactively redefine any of the four color maps. The standard screen set up provisions for background text and window colors can be retained if desired.

The specified color map can be subdivided into submaps where each submap will be comprised of a selected number of bits from the 8 available bits (the low order bit is bit 0).

Upon return from this routine the specified color map has been set up but the number of the map that is active has not been changed.

CALLING SEQUENCE: CALL CXMAP

SUBROUTINE CALLS: CBXSM

SOURCE CODE: &CXMAP

SUBROUTINE CXSAV

This routine saves the image stored in the IPIX array (in the HP memory) on the disk. The disk file name and an image identification label are entered interactively. The file must be a type 1 file with 128 words per record. The first record in the resulting file contains the 60 character image identification in the first 30 words, followed by NX and NY in words 31 and 32. The remaining records contain the pixel data with one pixel value per word. The image is stored row by row with no space between rows.

CALLING SEQUENCE: CALL CXSAV (NX, NY, ICRN)

WHERE:

NX - number of pixels per line IPIX array

NY - number of rows of pixels in IPIX array

ICRN - the desired cartridge number for disk file

SUBROUTINE CALLS: OPEN, CREAT, WRITF, CLOSE

SOURCE CODE: &CXSBI

NOTE: The image file can be read with a call to CBXGI or the system routine READF.

SUBROUTINE ERASE

This routine clears the screen and refills the screen with the specified color.

CALLING SEQUENCE: CALL ERASE (ICOL)

WHERE:

ICOL - the desired color (0-255)

SUBROUTINE CALLS: COLOR, CBXCM, WAIT

SOURCE CODE: &SBR

NOTE: When the default color map is active there is no color. In this case, the gray scale goes from 0 thru 255 where 0 = black and 255 = white.

SUBROUTINE HIST

This routine calculates and prints out a limited image HISTOGRAM. The image is modified according to HISTOGRAM stretching/compression, HISTOGRAM equalization, or thresholding. Either the image itself can be modified, or an additional color map is set up to effect the change. When the options of equalization and stretching/compression are used, the image itself is altered in the array IPIX for the option of thresholding color map 2 is affected.

CALLING SEQUENCE: COMMON/XPIX/IPIX (256,256)

```
      .
      .
      .
CALL HIST (IX, IY, NX, NY)
```

WHERE:

- IX - starting position in array, x direction
- IY - starting position in array, y direction
- NX - number of pixels per row, x direction
- NY - number of pixels per column, y direction

SUBROUTINE CALLS: CBXSM

SOURCE CODE: SCXHSM

SUBROUTINE IMAGE

This routine will output an image of up to 256 x 256 at the specified position on the screen.

CALLING SEQUENCE: COMMON/CXPIX/IPIX (256, 256)

```
      .
      .
      .
CALL IMAGE (NX, NY, ILX, ILY, NBITS, LBIT)
```

WHERE:

- NX - number of pixels per row in IPIX array
- NY - number of pixels per column in IPIX array
- ILX - desired screen x position of image
- ILY - desired screen y position of image
- NBITS - number of bits per pixel to be output
- LBIT - highest order bit of image byte to be used (DEFAULT = 7)
  IMAGE BYTE = (76543210)

SUBROUTINE CALLS: WINDO

SOURCE CODE: &CXSBI

NOTE: NX = 0, NY = 0 is considered the lower left corner of the screen. Negative values for NX and NY are also accepted, but the portion of the image which is located in the negative region is not displayed.

SUBROUTINE IMGCK

This routine can be used to verify that the image data array contains only valid intensities. The hardware limiting intensity values are 0 and 255, but other values may be set by the calling program. If any out of range intensity values are found, the user is given the opportunity to either compress or truncate the image data if desired.

CALLING SEQUENCE: CALL IMGCK (NX, NY, MIN, MAX, IERR)

WHERE:

NX - number of pixels per row

NY - number of rows of pixels

MIN - minimum desired pixel value (usually set to 0)

MAX - maximum desired pixel value (usually set to 255)

IERR - equals zero if all intensities are valid

SUBROUTINE CALLS: None

SOURCE CODE: &CXLB3

NOTE: MIN and MAX are specified by the calling program to add flexibility to this routine.

SUBROUTINE IMGNS

This routine adds Gaussian noise to the image in memory. The noise is stored on the disk with a mean = 0 and a standard deviation = 1. The noise level is adjusted as specified.

CALLING SEQUENCE: COMMON/CXBUF/RND (256), IDCB (144)
COMMON/CXPIX/IPIX (256, 256)
CALL IMGNS (NS, NY, SIGMA, AVG)

WHERE:

NX - number of pixels per row in IPIX array

NY - number of pixels per column in IPIX array

SIGMA - standard deviation

AVG - mean

SUBROUTINE CALLS: OPEN, CLOSE

SOURCE CODE: &CXLBS

SUBROUTINE INVRT

This routine will take the current image array and replace it with its inverse video representation. This mapping takes 0 → 255, 1 → 254,..., 255 → 0. The effect of this routine will not be seen on the CRT until the image is displayed.

CALLING SEQUENCE: CALL INVRT (NX, NY)

WHERE:

NX - number of points in X direction

NY - number of points in Y direction

SUBROUTINE CALLS: None

SOURCE CODE: &INVRT

NOTE: The array, IPIX, must contain image data upon entry to this routine. This routine is not compatible with the screen layout routine SETSC. Therefore routines should not be used from the same program. SETSC, however, provides its own inverse video capability.

SUBROUTINE IPLOT

This routine moves the cursor from its current position to the position specified. This move can be accomplished with the beam on or off. If the beam is on during the move, a line is drawn from the previous cursor position to the new cursor location.

CALLING SEQUENCE: CALL IPLOT (IX, IY, IPEN)

WHERE:

IX - new x coordinate of cursor (0-511)

IY - new y coordinate of cursor (0-479)

IPEN - =2 draw line (move with beam on)

=3 move with beam off

SUBROUTINE CALLS: CBXAD, CBXCM

SOURCE CODE: &SBR

SUBROUTINE MATUP

This routine will print out or alter a row of given matrix passed to it.

CALLING SEQUENCE: CALL MATUP (A, NR, NC, NAME)

WHERE:

A - matrix passed for printing
NR - number of rows
NC - number of columns
NAME - name of matrix passed

SUBROUTINE CALLS: None

SOURCE CODE: &CAUS1

NOTE: MATUP = MATRIX UPDATE

SUBROUTINE LIMAG

This routine uses linear interpolation between pixels to double the size of an image output to the screen.

CALLING SEQUENCE: COMMON/CXPIX/IPIX (256, 256)
COMMON/CXBUF/IBUF (4096)
CALL LIMAG (NX, NY, ILX, ILY, NBITS)

WHERE:

NX	-	number of pixels per row in IPIX array
NY	-	number of pixels per column in IPIX array
ILX	-	desired screen x position of image
ILY	-	desired screen y position of image
NBITS	-	number of bits per pixel to be output

SUBROUTINE CALLS: WINDO

SOURCE CODE: &LIMAG

SUBROUTINE SET

This routine allows the user to set up (or reset) a particular screen layout. Separate colors for background, text, and image window are defined. Positive and negative grayscale maps are set up in color maps 0 and 1 respectively, maps 2 and 3 are set up linearly in red and blue, and map 3 in red only.

This routine inquires after one has specified background, text, and window color if the chosen text color is adequate. If the user desires, it will display a sample and inquire if the user requests a change, if so, it will prompt for three new values (R, G, B) if not, the program will continue.

CALLING SEQUENCE: CALL SET (ILX, ILY, ISET)

WHERE:

ILX, ILY - lower left coordinates of image window

ISET - allows interactive setup (ISET=0) or
simply resets screen (ISET=1) or
resets screen with no previous setup (ISET= -1)

SUBROUTINE CALLS: CBXSM, ERASE, IPLOT, BOX, COLOR, TEXT

SOURCE CODE: &SET

NOTE: Intensities 0-252 inclusive are used for linear positive and negative grayscale maps. Intensities 253, 254, and 255 are used for window, text, and background color respectively. Each of these three colors are specified by its red, green, and blue intensity, (R-G-B). An image output on the screen must output all 8 pixel bits and must be scaled from 0-252 inclusive or color spots may appear in the black and white image.

SUBROUTINE STATS

This routine prints the status of the CBX terminal. The status word is printed in octal and has the following significant bits:

(sign bit)	bit 15	- CBX ready to accept data (Request A)
	bit 14	- Request B
	bit 2	- last command had not been completed
	bit 1	- last CBX parameter invalid
(lower order bit)	bit 0	- last CBX command invalid

CALLING SEQUENCE: CALL STATS (ISTAT)

WHERE:

ISTAT - the CBX status word

SUBROUTINE CALLS: PVON, CXSTS, PVOFF

SOURCE CODE: &SBR

NOTE: If STAT = -1 is returned the status could not be obtained. If this occurs, the CBX reset button should be pressed and the call remade.

SUBROUTINE STRIP

This routine uses a vector strip processor to filter the image. Each strip is filtered independently and the 2-D image data field is considered to be space invariant. The routine allows causal and noncausal filters and the strip vector format stresses horizontal image correlations more than vertical image correlations. The vector strip processor has the form of a matrix autoregressive (AR) model with potential look-ahead capability, and can be written as:

$$y(k) = A_{-1}y(k-1) + A_{-2}y(k-z) + \ldots + A_{-p}y(k-p) + A_{1}y(k+1)$$

$$+ \ldots A_{q}y(k+q) + Bu(k)$$

WHERE:

y(k) is a w-vector of filtered pixels
A is a w by w matrix
B is the input matrix
U(k) is the system input and where w is the strip width and p is the order of AR model

This model can be converted into state variable format as:

$$x(k+1) = Ax(k) + Ky(k)$$
$$Z(k) = Cx(k)$$

WHERE:

x(.) is the wp x 1 state vector
A is wp by wp system matrix and
K is wp by w gain matrix
y(.) is w x 1 input vector
Z(.) is w x 1 output vector
C is w by wp
$C = (I_w 0_w \ldots 0_w)$

CALLING SEQUENCE: STRIP (ISTRT, JSTRT, NXPTS, NYPTS)

WHERE:

NW	-	STRIP width
NP	-	number of past values used
NQ	-	number of future values used
N	-	order of state vector (equals NW*NP)
NQW	-	order of future vector (equals NQ*NP)

SUBROUTINE CALLS: MATUP

SOURCE CODE: &CAUS1

SUBROUTINE TEXT

This routine will print a string of characters on the screen at the specified location. The number of characters that can be passed with a single call is limited to a maximum of 60.

CALLING SEQUENCE: CALL TEXT (IX, IY, ITEXT, NCHAR)

WHERE:

IX	- the desired x coordinate for the lower left corner of text
IY	- the desired y coordinate
ITEXT	- the character string
NCHAR	- number of character to be output ($1 \leq$ NCHAR ≤ 60)

SUBROUTINE CALLS: CBXAD, CBXCM

SOURCE CODE: &SBR

SUBROUTINE WAIT

This routine will cause the HP to suspend operation of the program for a specified amount of time, up to a maximum of 59 seconds. This routine is called by the routine "ERASE" to insure that the CBX has completed its task before it is called again.

CALLING SEQUENCE: CALL WAIT (NSEC)

WHERE:

NSEC - the desired wait time in seconds (0<NSEC<60)

SUBROUTINE CALLS: EXEC

SOURCE CODE: &SBR

NOTE: This delay time is not exact. A requested delay of n seconds will result in an actual delay of up to one second less, but never greater than the requested delay.

SUBROUTINE WIND

This program provides the user with the option to define his own operating window, from which calculations can then be made.

CALLING SEQUENCE: CALL WIND (NX, NY ISTRT, JSTRT, N, M)

WHERE:

NX	- horizontal length of image in pixels
NY	- vertical length of image in pixels
ISTRT, JSTRT	- output variable representing the upper left corner of window
N	- horizontal length of window in pixels
M	- vertical length of window in pixels

SUBROUTINE CALLS: None

SOURCE CODE: &WINDW

SUBROUTINE WINDO

This routine transmits one 8K byte image window to or from the CBX. The image data is packed 2 bytes per word and is passed through the common block /CXBUF/. A description of image windows is contained in Appendix A.

CALLING SEQUENCE:

```
COMMON/CXBUF/IBUF(4096)
.
.
.
CALL WINDO (ICODE, NWIN, IERR)
```

WHERE:

ICODE	-	=0 read window (transmit from CBX to HP) =1 write window (transmit from HP to CBX)
NWIN	-	CBX window number (0-31)
IERR	-	error code (normal return =0)

SUBROUTINE CALLS: None

SOURCE CODE: &SBR

NOTE: This routine is generally not called directly from the user program.

Internal CBX 26-pin Floating Connector Pin #	CBX Function	External CBX 25-pin DB 25 Connector Pin #	HF Microcircuit Card Function	Hewlett Packard Connector Pin #
1	Ground	1	Ground	BB
2	Not Used	14	--	--
3	PDO0	2	Data In Bit 0	1
4	PDI0	15	Data Out Bit 0	A
5	PDO1	3	Data In Bit 1	2
6	PDI1	16	Data Out Bit 1	B
7	PDO2	4	Data In Bit 2	3
8	PDI2	17	Data Out Bit 2	C
9	PDO3	5	Data In Bit 3	4
10	PDI3	18	Data Out Bit 3	D
11	PDO4	6	Data In Bit 4	5
12	PDI4	19	Data Out Bit 4	E
13	PDO5	7	Data In Bit 5	6
14	PDI5	20	Data Out Bit 5	F
15	PDO6	8	Data In Bit 6	7
16	PDI6	21	Data Out Bit 6	H
17	PDO7	9	Data In Bit 7	8
18	PDI7	22	Data Out Bit 7	J
19	DT	10	Data Out Bit 14	S
20	RQA	23	Data In Bit 15	16
21	RQB	11	Data In Bit 14	15
22	NDR	24	Data Out Bit 15	T
23	Command	12	Data Out Bit 13	R
24	Not Used	25	—	--
25	Ground	13	Ground	24
26	Not Used			

TABLE A-1

The following is a list of the routines imbedded in the CBX driver program. The routines are FORTRAN callable and may be accessed by user programs. Care must be taken in their use or the HP system could become hung up.

PVON	turns privileged status on
PVOFF	turns privileged status off
STIN	reads Microcircuit Interface input register
COMND	outputs a single command byte (with handshakes)
CBXDO	outputs a string of parameters (with handshakes)
CXSTS	returns CBX status
RWIND	reads and packs one 8K byte window from CBX
WWIND	unpacks and writes one 8K byte window to CBX

TABLE A-2

CBX IMAGE MEMORY LAYOUT

Divided into 32 8K byte windows numbered from 0 thru 31 inclusive.

For Mode = 1 (512 X 480 X 8 bit image)

The even numbered windows affect the upper 256 lines. The odd numbered windows affect the lower 224 lines (Note: 480 lines visible)

Window 0

byte 1	Most significant bit (msb) of first 8 pixels
byte 2	Most significant bit of next 8 pixels of line 1
.	
.	
.	
byte 64	msb of last 8 pixels of line 1
byte 65	msb of first 8 pixels of line 3
byte 66	msb of next 8 pixels of line 3
.	
.	
.	
byte 8192	msb of last 8 pixels of line 255

Window 1	Contains msb of pixels in odd numbered lines from 257 - 479
Window 2	Contains msb of pixels in even numbered lines from 2 - 256
Window 3	Contains msb of pixels in even numbered lines from 258 - 480
Windows 4-7	Contain the next most significant bits of the same pixels referenced in windows 0-3.
.	
.	
.	
Windows 28-31	Contain the least significant bits of the same pixels referenced in windows 0-3.

TABLE A-3

DISTRIBUTION LIST

	No. of Copies
Commander US Army Tank-Automotive Command	
ATTN: Countermeasures Function, DRSTA-ZSC	20
Applied Research Function, DRSTA-ZSA	5
Technical Library, DRSTA-TSL	3
Warren, Michigan 48090	
Administrator Defense Technical Information Center ATTN: DTIC-DDA Cameron Station, Bldg. 5 Alexandria, Virginia 22314	10